# 실험 6. Multi-Tasking Box Drawing

## 1. 목적

이 실험에서는 실시간 운영체제가 제공하는 멀티태스킹 환경에서, 여러 개의 박스를 Nintendo DS의 위쪽 화면에 그리는 실험이다.

## 2. 실험 내용 설명

이 실험은 LCD Frame Buffer에 Pixel Value (color)를 write 함으로써 화면에 점을 찍고, 그 점으로 박스를 만드는 예를 본 뒤, 박스 여섯 개를 각각 다른 Timing으로 독립적인 Task로 단순하게 움직여보는 실험이다. 이 실험 결과는 다음 실험인 세마포를 이용한 동기화 실험에 기법에 그대로 사용될 예정이다.

## 3. 예제 소스

다음 소스는 Exp_Sample.c의 소스 일부로 닌텐도 DS의 위쪽 화면에 16x16 크기의 박스를 그리는 함수 이다.

```
22 #define COLOR_RED       RGB(31,  0,  0) /* Bright Red   */
23 #define COLOR_WHITE     RGB(31, 31, 31) /* Bright White */
24 #define COLOR_BLACK     RGB( 0,  0,  0)
25
26 #define BOX_WIDTH   16
27 #define BOX_HEIGHT  16
28 #define MAX_X       (SCREEN_WIDTH / BOX_WIDTH)
29 #define MAX_Y       (SCREEN_HEIGHT / BOX_HEIGHT)
30
31 void
32 draw_my_box(int pos_x, int pos_y, u16 color)
33 {
34     int i, j;
35     u32 *basePoint, pixel;
36
37     pixel = (color << 16) + color;
38     for (i = 0; i < BOX_HEIGHT; i++) {
39         basePoint = (u32 *)BG_GFX +
40           ((((pos_y * BOX_HEIGHT) + i) * SCREEN_WIDTH) + pos_x * BOX_WIDTH) / 2;
41         for (j = 0; j < (BOX_WIDTH / 2); j++)
42             *basePoint++ = pixel;
43     }
44 }
```

닌텐도의 LCD 화면 크기는 256 x 192 (SCREEN_WIDTH x SCREEN_HIGHT) 이며, 한 pixel은 16비트이며, RGB 각각 5 비트 (0-31)로 RGB 매크로로 컬러를 지정한다. 그리가 그리려는 박스는 16x16이므로, 박스는 화면을 16x12개 배치될 수 있다.

위 소스의 37번 라인은 한번에 두 pixel씩 write하기 위해, 16bit color를 32 비트로 확장한 것이다. 39번 라인에, BG_GFX는 위쪽 LCD의 시작 주소로, basePoint는 pos_x, pos_y가 가르키는 좌표의 왼쪽 상단 pixel의 주소가 된다. 그 38번 라인의 for loop는 그 위치에서부터 두 pixel 씩 (32bit) 한 줄을 찍고, 그 다음 줄로 basePoint를 옮긴 뒤 찍어나간다.

아래 소스는 아래쪽 LCD의 가상키를 읽어 상단 LCD의 해당 위치에 박스를 그린다. 키보드 관련한 내용은 지난 실험의 Key Queue를 이용한다. 라인 62-63은 이전의 박스를 지우고, 라인 64는 새로 입력된 위치에 박스를 그린다.

```
46 void
47 Exp_Sample(void)
48 {
49     u8 key, old_key = -1;
50
51     while (1) {
52         if (!kbhit()) {
53             if (NDS_SWITCH() & KEY_START)
54                 break;
55             vTaskDelay(30);
56             continue;
57         }
58         key = getkey();
59         if (key == old_key)
60             continue;
61
62         if (old_key >= 0)
63             draw_my_box(old_key, 8, COLOR_BLACK);    // Erase the Previous Box
64         draw_my_box(key, 8, COLOR_RED);              // Draw a New Box
65         old_key = key;
66     }
67     draw_my_box(old_key, 8, COLOR_BLACK);            // Erase the Previous Box
68     while (NDS_SWITCH() & KEY_START)
69         vTaskDelay(10);      // Wait while START KEY is being pressed
70 }
```

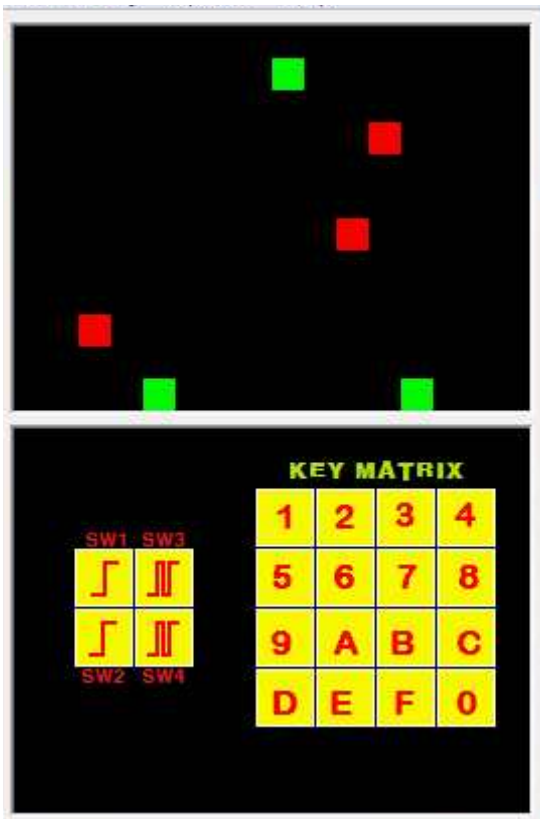
다음 소스는 예제 Task로 라인 66-68은 상단 LCD를 초기화 한다.

```
63 static
64 portTASK_FUNCTION(Exp_Task, pvParameters)
65 {
66     videoSetMode(MODE_5_2D);
67     vramSetBankA(VRAM_A_MAIN_BG);
68     bgInit(3,BgType_Bmp16, BgSize_B16_256x256, 0, 0);
69
70     while (1) {
71         Exp_Sample();
72     }
73 }
```

## 4. 응용 실험 (과제)

과제는 닌텐도 DS의 상단 LCD에 6개의 움직이는 박스를 그리는 실험이다. 실제 실행 화면은 아래와 같다. 위쪽 LCD에 빨강, 녹색 각각 3개의 박스가 있고, 빨간 박스는 좌우로, 녹색 박스는 상하로 움직인다. 각 박스는 독립적인 Task로 구성된다.

각 박스는 그려진 뒤 300msec 있다가 지우고, 임의의 시간동안 대기 한 후 한 칸을 상하좌우 등 진행 방향으로 이동한다. 벽을 만나면 방향을 바꾼다.



각 Task는 다음과 같은 Pseudo Code 방식으로 구현될 수 있다. 박스를 그리는 함수는 예제의 함수를 그대로 사용한다.

```
좌표_초기화;
while (1) {
    draw_box(현재_위치, color);
    delay(300msec)
    좌표_이동;
    벽을_만나면_방향_전환;
    delay(task별 다른 시간);
}
```

프로그램이 성공적으로 실행되면, 박스들이 상하 좌우로 움직이면서, 가끔은 녹색 박스와 빨간 박스가 겹치는 현상이 발생한다. 다음 실험은 세마포를 이용하여 이 현상을 없앤다.

모든 박스의 움직임이 같은 알고리즘에 의해 구현되기 때문에, 이런 프로그램은 같은 하나의 박스 움직이는 함수를 작성하고, 다른 부분 (방향, 시작 위치, 대기 시간)을 Task 시작의 Parameter로 전달하여 처리하면 간격하고 Debugging 하기 쉬운 프로그램이 된다. 다음 조각 소스들을 참조하라.

Task 마다 다른 부분을 parameter 구조로 global 선언:

```
29 struct parameters {
30     char *taskname;      // Task Name
31     int direction;       // Current Moving Direction
32     int basePoint;       // Starting Position
33     u32 color;           // Ball Color
34     int delay;           // Task Delay
35 };
36
37 struct parameters Param[NUM_TASK] = {
38     { "1", DIRECTION_RIGHT, 3,  COLOR_RED,     50 },
39     { "2", DIRECTION_RIGHT, 6,  COLOR_RED,     10 },
40     { "3", DIRECTION_RIGHT, 9,  COLOR_RED,    100 },
41     { "4", DIRECTION_DOWN,  4,  COLOR_GREEN,   20 },
42     { "5", DIRECTION_DOWN,  8,  COLOR_GREEN,   70 },
43     { "6", DIRECTION_DOWN,  12, COLOR_GREEN, 150 }
44 };
```

Task들을 다른 parameter를 전달하는 방식으로 Start:

```
86      for(i = 0, p = Param; i < NUM_TASK; i++, p++) {
87          xTaskCreate(Ball_Task, (const signed char *)(p->taskname), 1024,
88                  (void *)p, tskIDLE_PRIORITY + 5, NULL);
89      }
```

Task에서 Parameter를 받는 법:

```
92 static portTASK_FUNCTION(Ball_Task, pvParameters)
93 {
94     struct parameters *p = (struct parameters *)pvParameters;
```

## 5-1. 소스 및 샘플 프로그램

A. Template 소스는 자료실의 Ball.zip에 있음
B. 미리 만들어 놓은 Binary는 자료실의 Ball-Sample,nds, Ball-Homework.nds 임

## 5-2. 소스 및 샘플 프로그램 (Github 이용)

소스는 Github의 Ball 소스를 check-out하여 과제를 수행한다.

https://github.com/hl1itj/nds-ide/tree/microprocessor/lab/ball

에서, Checkout하여 과제를 수행하고 (main,c exp_sample.c 등 원하는 파일을 수정하고, Sample의 컴파일을 위해서는 지난번 KeyMatrix 구현 내용을 이용한다.)

(이 문서, 미리 build한 Ball-Sample,nds, Ball-Homework.nds는 doc directory에 있음)

https://github.com/hl1itj/nds-ide/tree/microprocessor/학번/ball

에서 각자 개발을 진행한다.